Leveraging the Margin More Carefully

Nir Krause KNIR@CS.HUJI.AC.IL
Yoram Singer SINGER@CS.HUJI.AC.IL
School of Computer Science and Engineering, The Hebrew University, Jerusalem, 91904, Israel

Abstract

Boosting is a popular approach for building accurate classifiers. Despite the initial popular belief, boosting algorithms do exhibit overfitting and are sensitive to label noise. Part of the sensitivity of boosting algorithms to outliers and noise can be attributed to the unboundedness of the margin-based loss functions that they employ. In this paper we describe two leveraging algorithms that build on boosting techniques and employ a bounded loss function of the margin. The first algorithm interleaves the expectation maximization (EM) algorithm with boosting steps. The second algorithm decomposes a non-convex loss into a difference of two convex losses. We prove that both algorithms converge to a stationary point. We also analyze the generalization properties of the algorithms using the Rademacher complexity. We describe experiments with both synthetic data and natural data (OCR and text) that demonstrate the merits of our framework, in particular robustness to outliers.

1. Introduction

In this paper we focus on the problem of supervised classification learning. In this setting we receive a training set of instance-label pairs $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^m$. For concreteness we assume that $\mathbf{x}_i \in \mathbb{R}^n$ and denote its j coordinate by x_{ij}. We confine this discussion to binary problems and thus $y_i \in \{-1, +1\}$. The class of classifiers we consider is the set of all threshold linear classifiers, $sign(\boldsymbol{\lambda} \cdot \mathbf{x})$ where $\boldsymbol{\lambda} \in I\!R^n$. We do not assume however that the data is linearly separable, thus there exist few examples $(\mathbf{x}_i, y_i) \in S$ such that $y_i(\boldsymbol{\lambda} \cdot \mathbf{x}_i) < 0$ for any $\boldsymbol{\lambda} \in I\!R^n$. The problem of finding a linear classifier that attains the minimal number of classification errors on S is NP-hard [10]. Therefore, learning techniques that employ a smooth and convex bound on the classification (0-1) error have been widely used. Specifically, boosting techniques use losses such as the exponential loss or the logistic-loss (see for instance [2] for a unified account of the two losses for boosting). The problem with convex losses is that they must diverge to ∞ as the margin of an example, $y_i(\boldsymbol{\lambda} \cdot \mathbf{x}_i)$ tends to $-\infty$. To overcome this problem, few approaches that employ *non-convex* losses have been proposed. Unfortunately, such methods do not yield a boosting algorithm in the classical notion of weak-learnability due to an impossibility theorem by Duffy and Helmbold [6]. Boosting-style methods that do not achieve a PAC-boosting property were referred to as *leveraging* algorithms by Duffy and Helmbold. Nonetheless, leveraging methods often yield effective and accurate classifiers and are robust to label noise.

In this work we propose two different leveraging algorithms that employ non-convex losses of the margin. The proposed algorithms are as simple to implement as the original AdaBoost algorithm of Freund and Schapire [9] and its closely related variants [15, 2]. As in [2], the two algorithms can be employed in a sequential manner in which a single feature (or weak-hypothesis) is chosen on each boosting step, as well as a fully parallel mode in which the weights of all the features are updated together. Last, but not least, when used in the classical sequential mode, the two algorithms induce a criterion (via an objective function) that guides the weak-learner in its search for a new hypothesis to add. This loss criterion is directly related to the decrease of the overall loss due to the addition of a new hypothesis.

The first algorithm is described in Sec. 2. We start by giving a probabilistic account which assumes the existence of an unobserved label noise. We derive an algorithm that combines the expectation maximization (EM) estimation procedure with boosting steps. This algorithm weights the examples according to their margin and their probability of being outliers. Boosting-style steps are used to update $\boldsymbol{\lambda}$ according to these weights. We prove that this algorithm converges to stationary point. The second approach, which is presented in Sec. 3, decomposes the non-convex loss function into a difference of two logistic functions, each of which is clearly convex. Here we derive a leveraging algorithm that is based on a recent additive update for boosting [3]. We analyze the loss functions that both algorithms employ in Sec. 4. We derive a generalization bound for both algorithms based on the Rademacher complexity in

Appearing in *Proceedings of the 21^{st} International Conference on Machine Learning*, Banff, Canada, 2004. Copyright 2004 by the author/owner.

Sec. 5. We conclude with experiments using synthetic and natural data that demonstrate the robustness of the algorithms to label noise. These experiments are described in Sec. 6.

Various previous research papers are used as building blocks in this paper. We use L. Baum's EM algorithm for maximum likelihood of incomplete data which was formally introduced in [4]. Our boosting steps are based on the boosting algorithms described in [2] and [3]. Our algorithmic approach of minimizing the difference of two convex functions is described thoroughly in [17] and was first employed for classification learning in [19] to devise robust support vector machines. The generalization analysis we employ is based on the generalization bounds given in [1]. Our work is also more remotely related to numerous papers in machine learning and statistics. The use of EM for missing labels and corrupted labels has already been mentioned in the discussion following the paper of Dempster, Laird, and Rubin. An example of an application of EM for missing labels can be found in [13]. The idea of combining EM with an iterative minimization procedure in a different context (and different analysis) was recently explored by Wang et al. [18]. There are many works that try to improve boosting, and give algorithms that are more robust to label noise. We survey here only a few of them. An algorithm that directly minimizes the 0-1 loss in k steps using weak learners with known accuracy is described in [7]. This idea is enhanced to an adaptive algorithm in [8], but using a complex algorithm. In [16] a PAC boosting algorithm is developed using smooth distributions. This algorithm can tolerate low malicious noise rates but it requires the access to a noise tolerant weak learner of a known accuracy. A similar approach to ours is presented by some papers. [12] employ the non-convex normalized sigmoid function as the loss function of their leveraging algorithm DOOM II. Their algorithm includes only a sequential version, and the objective function that the weak-learner has to minimize is not directly related to the decrease in the loss. Finally, the algorithm AdaBoost$_{reg}$ from [14], decreases the weight of examples that had large influence in previous rounds. It has extensions which solve the $\nu - LP$ problem (the LP equivalent of the SVM problem). These algorithms require line searches to compute the weights of the weak-learners whereas the algorithms described in this paper are simple to implement and their updates are computed analytically.

2. A Logistic Mixture Model

2.1. The Probabilistic Model

In our probabilistic model we assume that the noise-free (true) label t_i is generated by a logistic model that depends on $\mathbf{x}_i$ and a vector $\boldsymbol{\lambda} \in \mathbb{R}^n$. We further assume that the noise is a Bernoulli variable, with parameter ϵ, such that with probability ϵ the correct label is inverted, and the observed label y_i is obtained. The conditional probabilities due to these assumptions are as follows,

$$P_{(\boldsymbol{\lambda},\varepsilon)}(t_i|\mathbf{x}_i) = \frac{1}{1+e^{-t_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}} \ , \tag{1}$$

$$P_{(\boldsymbol{\lambda},\varepsilon)}(y_i, t_i|\mathbf{x}_i) = \begin{cases} \frac{1-\epsilon}{1+e^{-t_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}} & t_i = y_i \\ \frac{\epsilon}{1+e^{-t_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}} & t_i \neq y_i \end{cases} .$$

We define $\alpha_i(\boldsymbol{\lambda}, \varepsilon)$ as the probability that the observed label y_i is different from the noise-free label t_i. This is in fact the probability that label noise occurred given $\mathbf{x}_i$ and y_i,

$$\alpha_i(\boldsymbol{\lambda},\varepsilon) \overset{\text{def}}{=} P_{(\boldsymbol{\lambda},\varepsilon)}(t_i \neq y_i|\mathbf{x}_i, y_i) = \frac{\epsilon}{(1-\epsilon)e^{y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}+\epsilon}.$$

The goal is to devise a classifier that is more resistant to noise than boosting by incorporating the noise model above. We therefore cast our task as finding the parameters $\boldsymbol{\Theta} = (\boldsymbol{\lambda}, \epsilon)$ which minimize the negative log-likelihood of the observed data, which we call the logistic mixture loss,

$$\mathcal{L}_{LM}(\boldsymbol{\Theta}; S) = -\sum_{i=1}^{m} \ln\left(\frac{1-\epsilon}{1+e^{-y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}} + \frac{\epsilon}{1+e^{y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}}\right) \ .$$

In the algorithm description below and the experiments, we either allow ϵ to be fixed or estimate its value.

2.2. Combining Boosting Updates with EM

The EM algorithm [4] is the standard tool of choice for parameter estimation from incomplete data. The algorithm is composed of two steps: an E (Expectation) step and an M (Maximization) step. By repeating the E and M steps we converge to a stationary point of the log-likelihood of the model. We now specify the EM iterations for our settings. In the E step we replace each hidden variable (the unobserved value of t_i) with its expectation, i.e. $\alpha_i(\boldsymbol{\lambda}_t, \varepsilon_t)$, which we denote $\alpha_{i,t} \overset{\text{def}}{=} \alpha_i(\boldsymbol{\lambda}_t, \varepsilon_t)$ from here on.

In the M step we set $\boldsymbol{\Theta}_{t+1}$ to be the parameters that maximize the expectation of the log likelihood of the unobserved data and the observed data, where probabilities are calculated using $\boldsymbol{\Theta}_t$. That is we let $\boldsymbol{\Theta}_{t+1} = \text{argmax}_{\boldsymbol{\Theta}} Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}_t)$ where

$$\begin{aligned} Q(\boldsymbol{\Theta},\boldsymbol{\Theta}_t) &= \\ &= \sum_i \sum_{t_i} P_{\boldsymbol{\Theta}_t}(t_i|\mathbf{x}_i, y_i) \ln P_{\boldsymbol{\Theta}}(y_i, t_i|\mathbf{x}_i) \\ &= \sum_i \Big((1-\alpha_{i,t})\ln(1-\varepsilon) + \alpha_{i,t}\ln\varepsilon - \\ &\qquad (1-\alpha_{i,t})\ln(1+e^{-y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}) - \alpha_{i,t}\ln(1+e^{y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i})\Big) \ . \end{aligned}$$

Maximization of $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}_t)$ with respect to ε gives $\varepsilon_{t+1} = \frac{\sum_i \alpha_{i,t}}{m}$, where as $\boldsymbol{\lambda}_{t+1}$ is the minimizer of the loss,

$$\sum_i \Big((1-\alpha_{i,t})\ln(1+e^{-y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}) + \alpha_{i,t}\ln(1+e^{y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i})\Big) \ . \tag{2}$$

The above expression implies that maximizing $Q(\boldsymbol{\Theta}, \boldsymbol{\Theta}_t)$ with respect to $\boldsymbol{\lambda}$ is equivalent to minimizing a weighted logistic loss with $2m$ examples where each original example $(\mathbf{x}_i, y_i)$ is replaced with two weighted examples: $(\mathbf{x}_i, y_i)$ with weight $1 - \alpha_{i,t}$ and $(\mathbf{x}_i, -y_i)$ with weight $\alpha_{i,t}$.

```
LLM ($M, \varepsilon, T$, update-$\varepsilon, \mathcal{A}$)
  Requirements:
    $\mathcal{A} \subset \mathbb{R}^n_+$
    $\forall \mathbf{a} \in \mathcal{A} \; \sum_j a_j |M_{ij}| \leq 1$
    $\forall j \; \exists \mathbf{a} \in \mathcal{A} \; s.t. \; a_j > 0$
  init: $\boldsymbol{\lambda}_1 = 0, \; \varepsilon_1 = \varepsilon$
  For t=1,..,$T$
    $\forall i: \; \alpha_i = \frac{\varepsilon_t}{\varepsilon_t + (1-\varepsilon_t) e^{[M\boldsymbol{\lambda}_t]_i}} \;, \; q_i = \frac{1}{1+e^{[M\boldsymbol{\lambda}_t]_i}}$
    For j=1,...,n:
      $\tilde{W}_j^+ = \sum_{i:M_{ij}>0} (1-\alpha_i) q_i |M_{ij}|$
      $\tilde{W}_j^- = \sum_{i:M_{ij}<0} (1-\alpha_i) q_i |M_{ij}|$
      $W_j^+ = \tilde{W}_j^+ + \varepsilon_t \tilde{W}_j^- / (1-\varepsilon_t)$
      $W_j^- = \tilde{W}_j^- + \varepsilon_t \tilde{W}_j^+ / (1-\varepsilon_t)$
      $d_j = \frac{1}{2} \ln \left( \frac{W_j^+}{W_j^-} \right)$
    End
    $\mathbf{a} = \mathrm{argmax}_{\mathbf{a} \in \mathcal{A}} \sum_{j=1}^n a_j \left( \sqrt{W_j^+} - \sqrt{W_j^-} \right)^2$
    $\forall j \; \delta_j = a_j d_j$
    $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\delta}$
    if update-$\varepsilon$
      $\varepsilon_{t+1} = \frac{\sum_i \alpha_i}{m}$
    else
      $\varepsilon_{t+1} = \varepsilon_t$
End
```

Figure 1. The LLM algorithm.

An Approximate M-Step: We now focus on adapting a boosting-based algorithm for finding the minimizer of Eq. (2). Since there is no analytical solution for the minimizer, we need to devise an iterative minimization procedure. These iterations are interleaved with the E-step of EM which computes the auxiliary variables $\alpha_{i,t}$.

To derive our approximate M-step we first briefly describe one of the algorithms described by Collins et al. [2]. The algorithm belongs to a whole family of algorithms for finding the minimizer $\boldsymbol{\lambda}$ of the log-loss, $\sum_i \ln(1 + e^{-y_i \boldsymbol{\lambda} \cdot \mathbf{x}_i})$. To find the minimizer of the log-loss Collins et al. devised the following iterative procedure. Let $\boldsymbol{\lambda}_t$ denote the estimate for the minimizer of the log-loss at the j'th iteration. Then, each example $(\mathbf{x}_i, y_i)$ is assigned a weight q_i based on this estimate where, $q_i = \frac{1}{1+e^{y_i \boldsymbol{\lambda}_t \cdot \mathbf{x}_i}}$. The weight of an example reflects the probability of misclassifying the example according to the logistic model defined by Eq. (1). For ease of reading we define, from here on, the matrix M by $M_{i,j} = y_i x_{i,j}$. Based on the matrix M and the weights q_i we denote W_j^+ and W_j^- for each feature j,

$$W_j^+ = \sum_{i:M_{ij}>0} q_i |M_{ij}| \quad \text{and} \quad W_j^- = \sum_{i:M_{ij}<0} q_i |M_{ij}| \;.$$

Informally speaking, these weights reflect the correlation between the probability to misclassify the examples and the absolute value of the j'th feature for each example. The next estimate $\lambda_{t+1,j}$ is computed from the current estimate and the weights as follows,

$$\lambda_{t+1,j} = \lambda_{t,j} + 1/2 \ln(W_j^+ / W_j^-) \;. \tag{3}$$

The above update was derived for noise-free classification settings with the log-loss. Our setting is however more involved. We observe a noisy version of the labels and need to minimize the loss with respect to the, hidden, true labels. Therefore, each example is associated with two possible true labels $y_i, -y_i$, with probabilities $1 - \alpha_{i,t}, \alpha_{i,t}$ respectively. For each possible label we need to take into account its weight and probability according to the logistic model (Eq. (1)) as reflected by q_i. The end result is that to compute W_j^+ and W_j^- we need to go over all indices i and "split" the contribution of each example between W_j^+ and W_j^-. This yields the following calculation for the loss defined by Eq. (2),

$$\begin{aligned} W_j^+ &= \sum_{i:M_{ij}>0} (1-\alpha_{i,t}) q_i |M_{ij}| \qquad (4) \\ &\quad + \sum_{i:M_{ij}<0} \alpha_{i,t} (1-q_i) |M_{ij}| \\ W_j^- &= \sum_{i:M_{ij}<0} (1-\alpha_{i,t}) q_i |M_{ij}| \qquad (5) \\ &\quad + \sum_{i:M_{ij}>0} \alpha_{i,t} (1-q_i) |M_{ij}| \;. \end{aligned}$$

We can now proceed as before to compute $\boldsymbol{\lambda}_{t+1}$ from $\boldsymbol{\lambda}_t$ as given by Eq. (3). The M-step of an EM algorithm requires finding the minimizer of Eq. (2). This can easily be done using many iterations of the above update rule. However, as we show in the next section, it is sufficient to perform a single update as an approximate M-step. In this case we only update $\boldsymbol{\Theta}'$ such that it decreases $Q(\boldsymbol{\Theta}', \boldsymbol{\Theta})$, yet not minimizes it, and this is in fact a GEM algorithm. When performing a single approximate M-step we can further simplify the boosting-based update. In which case we can rewrite Eq. (4) and Eq. (6) and avoid the additional computation imposed when computing W_j^+ and W_j^-. This is due to the fact that the same value of $\boldsymbol{\lambda}$ and ε is employed for the computation of both $\alpha_{i,t}$ and q_i. We now describe the more efficient single approximate M-step. Define $\tilde{W}_j^+ = \sum_{i=1:M_{ij}>0}^m (1-\alpha_{i,t}) q_i |M_{ij}|$ and $\tilde{W}_j^- = \sum_{i=1:M_{ij}<0}^m (1-\alpha_{i,t}) q_i |M_{ij}|$. For fixed $\boldsymbol{\lambda}$ and ε it is easy to verify that $(1-\alpha_{i,t}) q_i = \frac{1-\varepsilon}{\varepsilon} \alpha_{i,t} (1-q_i)$. Hence we can rewrite the original variables W_j^+ and W_j^- as, $W_j^+ = \tilde{W}_j^+ + \frac{\varepsilon}{1-\varepsilon} \tilde{W}_j^-$ and $W_j^- = \tilde{W}_j^- + \frac{\varepsilon}{1-\varepsilon} \tilde{W}_j^+$. The pseudo code for the version that alternates between an E-step and a single boosting-based M-step is given in Fig. 1. We term the algorithm the Leveraging Logistic Mixture algorithm (LLM). This version was used in our experiments, however other variants, such as extensions to multiclass, can also be derived. The details of these variants and extensions are omitted due to the lack of space.

A parametric family of algorithms: This algorithm receives as input a template set $\mathcal{A}$. Using the template set we describe in a unified view both a parallel algorithm and a sequential one. The choice of algorithm to be implemented is done by setting the appropriate template set. By setting $\mathcal{A} = \{(1, 1, ..., 1)\}$, we obtain a parallel algorithm - an algorithm that updates all the indices in each step. If we let $\mathcal{A} = \{e_i\}_{i=1}^n$ (e_i denotes the ith unit vector), we receive a sequential boosting algorithm. This algorithm has a simple criterion for choosing the weak learner at each step - choose the weak learner j which maximizes $\left(\sqrt{W_j^+} - \sqrt{W_j^-}\right)^2$. Our proof of convergence also holds if we change $\mathcal{A}$ in each iteration, as long as the constraints defined on $\mathcal{A}$ and M are kept satisfied. Therefore we can also implement a combined version, in which we sequentially update, yet once in a while we update the weight of the weak learners accumulated so far in parallel.

2.3. Convergence Analysis

In this section we discuss the convergence of the LLM algorithm. In short, the LLM algorithm belongs to the class of Generalized EM algorithms [4]. The following lemma shows that each approximate M-step is guaranteed to increase the value of the auxiliary function Q. Our proof combines the proof techniques from [2] with standard analysis for EM.

Lemma 2.1: *Let $\mathbf{\Theta}'$ be the set of parameters obtained after the update of $\boldsymbol{\lambda}$. Then $Q(\mathbf{\Theta}', \mathbf{\Theta}) \geq Q(\mathbf{\Theta}, \mathbf{\Theta})$, with equality only if $\mathbf{\Theta}$ is a stationary point of Q.*

Proof: To prove the theorem we view the loss as induced by a sample of $2m$ examples where the first m examples are $(\mathbf{x}_i, y_i)$ with weights $\beta_i = 1 - \alpha_i(\boldsymbol{\lambda}, \varepsilon)$ and the next m examples are $(\mathbf{x}_i, -y_i)$ (denoted $\mathbf{x}_{i+m} = \mathbf{x}_i,\ y_{i+m} = -y_i$) with weights $\beta_{i+m} = \alpha_i(\boldsymbol{\lambda}, \varepsilon)$. Given this representation we can rewrite Q as,

$$Q(\mathbf{\Theta}', \mathbf{\Theta}) = c(\varepsilon') - \sum_{i=1}^{2m} \beta_i \ln(1 + e^{-y_i \boldsymbol{\lambda}' \cdot \mathbf{x}_i}) \ ,$$

where c is a constant that depends only on ε'.

We denote by q_i and q_i' the weight of example i before and after the update respectively, $q_i = \frac{1}{1+e^{[M\boldsymbol{\lambda}]_i}}$ and $q_i' = \frac{1}{1+e^{[M\boldsymbol{\lambda}']_i}} = \frac{1}{1+e^{[M(\boldsymbol{\lambda}+\boldsymbol{\delta})]_i}}$. From these equations we get that, $q_i' = \frac{q_i e^{-[M\boldsymbol{\delta}]_i}}{1-q_i+q_i e^{-[M\boldsymbol{\delta}]_i}}$. Using this equality and the inequality $1 + x \leq e^x$, we get that,

$$\ln(1 - q_i') - \ln(1 - q_i) \geq q_i(1 - e^{-[M\boldsymbol{\delta}]_i}) \ . \qquad (6)$$

We can rewrite $Q(\mathbf{\Theta}', \mathbf{\Theta})$ using q_i' as follows,

$$Q(\mathbf{\Theta}', \mathbf{\Theta}) = c(\varepsilon') + \sum_{i=1}^{2m} \beta_i \ln(1 - q_i') \ . \qquad (7)$$

Note also that using β_i we can rewrite the weights W_j^+ and W_j^- as follows,

$$W_j^+ = \sum_{i:M_{ij}>0} \beta_i q_i |M_{ij}| \ ; \ W_j^- = \sum_{i:M_{ij}<0} \beta_i q_i |M_{ij}| \ . \qquad (8)$$

Denoting $s_{ij} = sign(M_{ij})$, we can lower bound the change in Q in the M-step as follows,

$$\begin{aligned}
&Q(\mathbf{\Theta}', \mathbf{\Theta}) - Q(\mathbf{\Theta}, \mathbf{\Theta}) \\
&\overset{\text{Eq. (7)}}{=} \sum_{i=1}^{2m} \beta_i \left[\ln(1 - q_i') - \ln(1 - q_i)\right] \\
&\overset{\text{Eq. (6)}}{\geq} \sum_{i=1}^{2m} \beta_i q_i \left[1 - e^{-[M\boldsymbol{\delta}]_i}\right] \\
&= \sum_{i=1}^{2m} \beta_i q_i \left[1 - \exp(-\sum_{j=1}^{n} a_j d_j s_{ij} |M_{ij}|)\right] \\
&\geq \sum_{i=1}^{2m} \beta_i q_i \left[\sum_{j=1}^{n} a_j |M_{ij}| (1 - e^{d_j s_{i,j}})\right] \qquad (9) \\
&\overset{\text{Eq. (8)}}{=} \sum_{j=1}^{n} a_j \left(W_j^+ + W_j^- - W_j^+ e^{d_j} - W_j^- e^{-d_j}\right) \\
&= \sum_{j=1}^{n} a_j \left(\sqrt{W_j^+} - \sqrt{W_j^-}\right)^2 \qquad (10) \\
&\overset{\text{def}}{=} A(\mathbf{q}) \ .
\end{aligned}$$

Eq. (9) follows from Jensen's inequality applied to the convex function e^x while using the condition that $\sum_j a_j |M_{ij}| \leq 1$; to get Eq. (10) we use $d_j = \frac{1}{2} \ln(W_j^+ / W_j^-)$ as in the algorithm.

The function $A(\mathbf{q})$ is an auxiliary function that bounds from below the increase of Q. Clearly, $A(\mathbf{q}) \geq 0$ and $A(\mathbf{q}) = 0$ iff $\forall j \ \ W_j^+ = W_j^-$. When $A(\mathbf{q}) = 0$ we get by simple derivation that $\frac{\partial}{\partial \lambda_j'} Q(\mathbf{\Theta}, \mathbf{\Theta}) = W_j^+ - W_j^- = 0$, this means that $\boldsymbol{\lambda}'$ is a stationary point of Q. If we have not reached a stationary point the value of Q will continue to increase on each approximate M-step. ∎

Using the definition of Q and the above lemma we see that

$$Q(\boldsymbol{\lambda}_{t+1}, \boldsymbol{\lambda}_{t+1}) \geq Q(\boldsymbol{\lambda}_{t+1}, \boldsymbol{\lambda}_t) > Q(\boldsymbol{\lambda}_t, \boldsymbol{\lambda}_t) \ .$$

Thus, the series $Q(\boldsymbol{\lambda}_t, \boldsymbol{\lambda}_t)$ is monotonically increasing and converges in value to a stationary point. It directly follows that the loss, $\mathcal{L}_{LM}$, converges in value to a stationary point.

3. A Logistic Difference Model

In this section we present a seemingly different approach to obtain a bounded margin loss function and a corresponding leveraging algorithm. Instead of devising a mixture model of logistic function we devise a bounded margin loss by taking the *difference* of two logistic functions as follows. Let z denote the margin of an example. The logistic loss of an example with margin z is $\ell(z) = \ln(1 + e^{-z})$. Let μ be a positive constant. A "shifted" version of the loss is obtained by

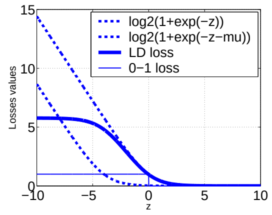

Figure 3. The construction of the LD loss ($\mu = 4$).

```
LLD($M, \mu, T, \mathcal{A}$)
  Requirements:
    $\mathcal{A} \subset \mathbb{R}_+^n$
    $\forall \mathbf{a} \in \mathcal{A}: \sum_{i,j} a_j M_{i,j}^2 \leq 2$
    $\forall j\ \exists \mathbf{a} \in \mathcal{A}\ s.t.\ a_j > 0$
  init: $\boldsymbol{\lambda}_1 = 0$
  For t=1,..,$T$
    $q_i = \frac{1}{1+e^{[M\boldsymbol{\lambda}_t]_i}}, \quad g_i = \frac{1}{1+e^{[M\boldsymbol{\lambda}_t]_i+\mu}}$
    $W_j = \sum_i M_{i,j}(q_i - g_i)$
    $a = \text{argmax}_{a \in \mathcal{A}} \sum_j a_j W_j^2$
    $\forall j\ \delta_j = a_j W_j$
    $\boldsymbol{\lambda}_{t+1} = \boldsymbol{\lambda}_t + \boldsymbol{\delta}$
```

Figure 2. The LLD Algorithm.

adding μ to z, $\ell(z+\mu) = \ln(1+e^{-z-\mu})$. We now define the loss at margin z to be the difference between the logistic loss and its shifted version,

$$\ln(1+e^{-z}) - \ln(1+e^{-z-\mu}) \ . \tag{11}$$

The construction of this loss is described in Fig. 3. We term this loss the logistic-difference (LD) loss. As in the logistic mixture model, we would like to find a vector $\boldsymbol{\lambda}$ such that the sign of $\boldsymbol{\lambda} \cdot \mathbf{x}_i$ is equal to y_i as much as possible. Thus, the logistic-difference loss of the sample S w.r.t $\boldsymbol{\lambda}$ is,

$$\begin{aligned}\mathcal{L}_{LD}(\boldsymbol{\lambda}; S) &= \sum_{i=1}^m \Big(\ell(y_i(\boldsymbol{\lambda}\cdot\mathbf{x}_i)) - \ell(y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i + \mu)\Big) \\ &= \sum_{i=1}^m \Big(\ln(1+e^{-y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i}) - \ln(1+e^{-y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i-\mu})\Big) \ .\end{aligned}$$

Our leveraging algorithm for the LD loss is based on a recent *additive* update for boosting [3]. In our algorithm, each example is given a weight which is equal to minus the derivative of the loss evaluated at the example's margin. More formally, let $g_\mu(z) = -\frac{d}{dz}\ell(z+\mu) = \frac{1}{1+e^{z+\mu}}$ then the weight of an example attaining a margin $z_i = y_i(\boldsymbol{\lambda}\cdot\mathbf{x}_i)$ is $g_0(z_i) - g_\mu(z_i)$. The pseudo code of the resulting leveraging algorithm, called the Leveraging Logistic Difference algorithm (LLD), is given in Fig. 2. Intuitively, on each leveraging iteration, LLD approximates the concave part of the LD loss, $-\ell(z+\mu)$, with a linear function. Since the difference between a convex function and a linear function is convex, we can now use boosting technology. Specifically, LLD performs an additive boosting-step, analogous to the one described in [3] on the convexified function. We also exploit this construction to derive in the sequel a lower bound on the progress of LLD and prove its convergence.

Convergence analysis Our convergence proof is divided into two parts. In the first part we show a simple convex upper bound on the logistic difference loss. In the second part we show that the t'th iteration decreases this upper bound (unless we reached a stationary point). We get that the logistic difference loss decreases on each iteration, and therefore converges to a stationary point as it is bounded below.

Recall that every differentiable concave function f is bounded by its affine minorization at every point, i.e.

$$f(\boldsymbol{\lambda}) \leq f(\boldsymbol{\lambda}_t) + (\boldsymbol{\lambda} - \boldsymbol{\lambda}_t)\cdot\nabla f(\boldsymbol{\lambda}_t) \ .$$

By applying this to the concave part of the loss we get

$$\begin{aligned}&-\sum_{i=1}^m \ell(y_i\boldsymbol{\lambda}_{t+1}\cdot\mathbf{x}_i + \mu) \\ &\leq \ -\sum_{i=1}^m \Big(\ell(y_i\boldsymbol{\lambda}_t\cdot\mathbf{x}_i+\mu) + \\ &\qquad (\boldsymbol{\lambda}-\boldsymbol{\lambda}_t)\cdot\nabla_{\boldsymbol{\lambda}_t}\ell(y_i\boldsymbol{\lambda}_t\cdot\mathbf{x}_i+\mu)\Big) \\ &= \ \eta(\boldsymbol{\lambda}_t) + \boldsymbol{\lambda}\cdot\sum_{i=1}^m g_\mu(y_i\boldsymbol{\lambda}_t\cdot\mathbf{x}_i)y_i\mathbf{x}_i \ ,\end{aligned}$$

where $\eta(\boldsymbol{\lambda}_t) = \sum_{i=1}^m(\boldsymbol{\lambda}_t\cdot g_\mu(y_i\boldsymbol{\lambda}_t\cdot\mathbf{x}_i)y_i\mathbf{x}_i - \ell(y_i\boldsymbol{\lambda}_t\cdot\mathbf{x}_i+\mu))$. This upper bound is linear and thus also convex. We add it to the convex logistic loss and get the following convex bound on the loss,

$$\begin{aligned}\hat{L}(\boldsymbol{\lambda},\boldsymbol{\lambda}_t;S) &= \eta(\boldsymbol{\lambda}_t) + \\ &\quad \sum_{i=1}^m \Big(\ell(y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i) + \boldsymbol{\lambda}\cdot g_\mu(y_i\boldsymbol{\lambda}_t\cdot\mathbf{x}_i)y_i\mathbf{x}_i\Big) \\ &= \eta(\boldsymbol{\lambda}_t) + \boldsymbol{\lambda}\cdot\mathbf{d} + \sum_{i=1}^m \ell(y_i\boldsymbol{\lambda}\cdot\mathbf{x}_i) \ ,\end{aligned}$$

where $\mathbf{d} = \sum_{i=1}^m g_\mu(y_i\boldsymbol{\lambda}_t\cdot\mathbf{x}_i)y_i\mathbf{x}_i$. We deduce that for all $\boldsymbol{\lambda}$: $\mathcal{L}_{LD}(\boldsymbol{\lambda};S) \leq \hat{L}(\boldsymbol{\lambda},\boldsymbol{\lambda}_t;S)$, with equality for $\boldsymbol{\lambda} = \boldsymbol{\lambda}_t$.

We now concentrate our efforts on showing that $\hat{L}(\boldsymbol{\lambda},\boldsymbol{\lambda}_t;S)$ decreases unless we are at a stationary point.

Lemma 3.1: *The decrease in $\hat{L}(\boldsymbol{\lambda}_{t+1},\boldsymbol{\lambda}_t;S)$ satisfies,*

$$\Delta_t = \hat{L}(\boldsymbol{\lambda}_t,\boldsymbol{\lambda}_t;S) - \hat{L}(\boldsymbol{\lambda}_{t+1},\boldsymbol{\lambda}_t;S) \geq \frac{1}{2}\sum_{j=1}^n a_j W_j^2$$

Proof: We prove the lemma by using a quadratic function $Q(\alpha)$ which upper bounds $\hat{L}(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\Lambda},\boldsymbol{\lambda}_t;S)$ (for any fixed $\boldsymbol{\lambda}_t,\boldsymbol{\Lambda}$) and then prove progress w.r.t this upper bound. Define,

$$Q(\alpha) = \hat{L}(\boldsymbol{\lambda}_t,\boldsymbol{\lambda}_t;S) + (\nabla L_0(\boldsymbol{\lambda};S) + \mathbf{d})\cdot\boldsymbol{\Lambda}(\alpha - \frac{\alpha^2}{2})$$

In order to prove that Q upper bounds $\hat{L}$ we define their difference $\Gamma(\alpha) = Q(\alpha) - \hat{L}(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\Lambda},\boldsymbol{\lambda}_t;S)$ and show that $\Gamma(\alpha) \geq 0$. We do this by showing that Γ is convex and its minimum is attained at $\alpha = 0$ (where we have $\Gamma(0) = 0$). First note that $\Gamma'(0) = 0$. Hence, 0 is a stationary point of Γ. Taking the second derivative of Γ we get, $\Gamma''(\alpha) = -(\nabla L_0(\boldsymbol{\lambda};S) + \mathbf{d})\cdot\boldsymbol{\Lambda} - \boldsymbol{\Lambda}^T H\boldsymbol{\Lambda}$, where $H = \sum_{i=1}^m L_0''(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\lambda}_t;S)\mathbf{x}_i\mathbf{x}_i^T$. Also, routine calculations yield that $0 \leq L_0''(\boldsymbol{\lambda}_t + \alpha\boldsymbol{\lambda}_t;S) \leq 1/2$ and hence

$$\Gamma''(\alpha) \geq -(\nabla L_0(\boldsymbol{\lambda};S) + \mathbf{d})\cdot\boldsymbol{\Lambda} - \frac{1}{2}\sum_{i=1}^m(\boldsymbol{\Lambda}\cdot\mathbf{x}_i)^2 \ .$$

Now we plug in the definitions we use in the algorithm. We note that $\mathbf{W} = -(\nabla L_0(\boldsymbol{\lambda};S) + \mathbf{d})$ and $\Lambda_j = \delta_j = a_j W_j$.

$$\begin{aligned}\Gamma''(\alpha) &\geq \sum_{j=1}^{n} a_j W_j^2 - \frac{1}{2}\sum_{i=1}^{m}(\sum_{j=1}^{n}\sqrt{a_j}W_j\sqrt{a_j}\mathbf{x}_{i,j})^2 \\ &\geq \sum_{j=1}^{n} a_j W_j^2 - \frac{1}{2}\sum_{i=1}^{m}(\sum_{j=1}^{n} a_j W_j^2)(\sum_{k=1}^{n} a_k \mathbf{x}_{i,k}^2) \\ &= \sum_{j=1}^{n} a_j W_j^2 (1 - \frac{1}{2}\sum_{i=1}^{m}\sum_{k} a_k \mathbf{x}_{i,k}^2) \geq 0 \ .\end{aligned}$$

Where we used Cauchy-Schwartz inequality in the first inequality, and the constraint $\sum_{\mathbf{x}\in S}\sum_{k=1}^{n} a_k \mathbf{x}_k^2 \leq 2$ in the last inequality.

We have shown that $\Gamma(\alpha) \geq 0$ and thus Q upper bounds $\hat{L}$. Since $Q(0) = \hat{L}(\boldsymbol{\lambda}_t, \boldsymbol{\lambda}_t; S)$ we can bound the progress Δ_t from below as follows,

$$\begin{aligned}\Delta_t &= \hat{L}(\boldsymbol{\lambda}_t, \boldsymbol{\lambda}_t; S) - \hat{L}(\boldsymbol{\lambda}_{t+1}, \boldsymbol{\lambda}_t; S) \\ &\geq Q(0) - Q(1) = \frac{1}{2}\sum_{j=1}^{n} a_j W_j^2 \ .\end{aligned}$$ ∎

Corollary 3.2: *The decrease in the loss at step t satisfies*

$$\begin{aligned}\mathcal{L}_{LD}(\boldsymbol{\lambda}_t; S) - \mathcal{L}_{LD}(\boldsymbol{\lambda}_{t+1}; S) &\geq \\ \hat{L}(\boldsymbol{\lambda}_t, \boldsymbol{\lambda}_t; S) - \hat{L}(\boldsymbol{\lambda}_{t+1}, \boldsymbol{\lambda}_t; S) &\geq \frac{1}{2}\sum_{j=1}^{n} a_j W_j^2\end{aligned}$$

Lemma 3.3: *The algorithm converges by value to a stationary point of its loss.*

Proof: As long as $\nabla\mathcal{L}_{LD} \neq 0$ we decrease the loss by a positive amount (because $W = \nabla\mathcal{L}_{LD}$). Therefore the sequence of losses decreases. It is bounded below by 0, and hence it must converge. By continuity the convergence point must be a point in which $\nabla\mathcal{L}_{LD} = 0$. ∎

4. A unified view of the loss functions

In this section we show that there is a simple bijection from ϵ to μ which makes the losses of LLM and LLD identical. We also describe a simple affine transformation that normalizes either loss so that the resulting loss approaches 0 as the margin goes to $+\infty$ while bounding everywhere the $0-1$ loss. Let us fix ϵ and μ. We now add a constant to each loss and divide the result by another constant in order to normalize the losses. The resulting normalized losses are,

$$\begin{aligned}\mathcal{L}_{LM}^{N}(z) &= \frac{-\ln\left(\frac{1-\epsilon}{1+e^{-z}} + \frac{\epsilon}{1+e^{z}}\right) + \ln(1-\epsilon)}{\ln(2) + \ln(1-\epsilon)} \qquad (12) \\ \mathcal{L}_{LD}^{N}(z) &= \frac{\ln(1+e^{-z}) - \ln(1+e^{-z-\mu})}{\ln(2) - \ln(1+e^{-\mu})}\end{aligned}$$

Examining carefully the above losses, we can see that by setting $e^{-\mu} = \frac{\epsilon}{1-\epsilon}$ and further algebraic manipulations we obtain that,

$$\begin{aligned}\mathcal{L}_{LD}^{N}(z) &= \frac{\ln\left(\frac{1+e^{-z}}{1+e^{-z-\mu}}\right)}{\ln(2) - \ln(1+e^{-\mu})} \\ &= \frac{\ln\left(\frac{1-\epsilon}{\frac{1-\epsilon}{1+e^{-z}} + \frac{\epsilon}{1+e^{z}}}\right)}{ln(2) + \ln(1-\epsilon)} = \mathcal{L}_{LM}^{N}(z) \ .\end{aligned}$$

Hence, the two losses are identical subject to the transformation of variables, $\mu = \ln(\frac{1-\epsilon}{\epsilon})$, or equivalently $\epsilon = \frac{1}{1+e^{\mu}}$. Since the two losses are equivalent, from now on we refer in our analysis only to the logistic mixture loss.

For later use we calculate the Lipschitz constant of the loss, that is, find a constant ζ such that for any z, z', $|L(z) - L(z')| \leq \zeta|z - z'|$. For $\mathcal{L}_{LM}^{N}(z)$ this constant is $\zeta = \frac{1-2\varepsilon}{(\ln(2)+\ln(1-\varepsilon))(1+2\sqrt{\varepsilon(1-\varepsilon)})}$. We also note that $\mathcal{L}_{LM}^{N}(z)$ is bounded above by $M = \frac{\ln\frac{1-\varepsilon}{\varepsilon}}{\ln(2)+\ln(1-\varepsilon)}$.

5. Generalization analysis

In [1] generalization bounds are given for classes of functions using their Rademacher complexity. In short, the Rademacher complexity of a class of functions F, denoted $R_m(F)$, is defined as, $R_m(F) = \mathbf{E}\hat{R}_m(F)$ where

$$\hat{R}_m(F) = \mathbf{E}\left[\sup_{f\in F}\left|\frac{2}{m}\sum_{j=1}^{m}\sigma_i f(X_i)\right| |X_1, ..., X_m\right] \ ,$$

and σ_i are independent random variables sampled uniformly from $\{\pm 1\}$. The bounds in [1] are suited for decision theoretic settings in which we attempt to minimize a combinatorial loss function (such as the classification error) via the minimization of a dominating cost function whose range is $[0, 1]$. In our case the loss function is the classification error and the dominating cost function is the loss $\mathcal{L}_{LM}^{N}(z)$ defined in Eq. (12), whose range is $[0, M]$. In order to use the Rademacher bounds from [1] we need to slightly generalize Thm. 8 of [1] to cost functions whose range is $[0, M]$.

Theorem 5.1: [Bartlett and Mendelson, Thm. 8] Consider a loss function $\mathcal{L} : \mathcal{Y} \times A \to [0, M]$ and a dominating cost function $\phi : \mathcal{Y} \times A \to [0, M]$. Let F be a class of functions mapping from X to A and let $(X_i, Y_i)_{i=1}^{m}$ be independently selected according to the probability measure P. Then, for any integer m and any $0 < \delta < 1$ with probability at least $1 - \delta$ over samples of length m, every f in F satisfies

$$E\mathcal{L}(Y, f(x)) \leq \hat{E}_m\phi(Y, f(x)) + R_m(\tilde{\phi}\circ F) + 4M\sqrt{\frac{\ln(2/\delta)}{2m}}$$

where $\tilde{\phi}\circ F = \{(x, y) \mapsto \phi(y, f(x)) - \phi(y, 0) : f \in F\}$.

To derive a bound for our problem, we need to bound the Rademacher complexity of linear classifiers. Bartlett and Mendelson proved a bound on kernel functions. This bound can be rewritten in our setting and notation as follows: Assume $||\mathbf{x}_i||_2 \leq 1$ and let $F = \{x \mapsto \boldsymbol{\lambda}\cdot x|\ ||\boldsymbol{\lambda}||_2 \leq B\}$, then $\hat{R}_m(F) \leq \frac{2B}{\sqrt{m}}$. In the algorithm we assumed that $||\mathbf{x}_i||_1 \leq 1$, from which follows that $||\mathbf{x}_i||_2 \leq 1$. Using standard regularization techniques (as described in [3]) implies that $||\boldsymbol{\lambda}||_1 \leq C$. This means that $||\boldsymbol{\lambda}||_2 \leq ||\boldsymbol{\lambda}||_1 \leq C$. Combining these norm inequalities, we get that in our case $\hat{R}_m(F) \leq \frac{2C}{\sqrt{m}}$.

We conclude by adapting the result of theorem 21 in [1] to our setting. Using the bound M and Lipschitz constant ζ

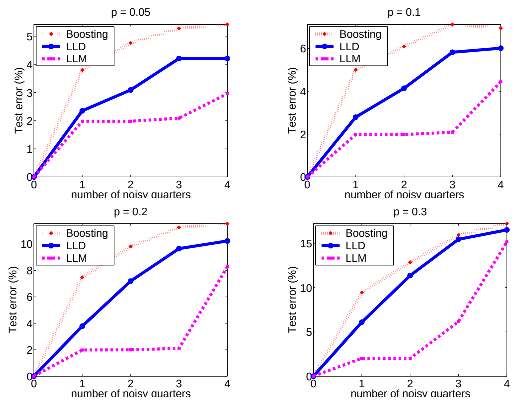

Figure 4. Comparison of boosting, LLM and LLD on synthetic data with different noise rates.

which were calculated in Sec. 4 for our cost function $\mathcal{L}_{LM}^N$, and by applying Theorem 5.1 we get,

Corollary 5.2: *With probability* $1-\delta$ *every* $f \in F$ *satisfies*

$$P(Yf(x) \leq 0) \leq \hat{E}_m \mathcal{L}_{LM}^N(Y, f(x)) + 8\frac{\ln\frac{1-\varepsilon}{\varepsilon}}{\ln(2)+\ln(1-\varepsilon)}\sqrt{\frac{\ln(4/\delta)}{2m}} + \frac{4B(1-2\varepsilon)}{(\ln(2)+\ln(1-\varepsilon))(1+2\sqrt{\varepsilon(1-\varepsilon)})\sqrt{m}}$$

6. Experiments

Synthetic data: In this experiment we generated 1000 random points in $I\!R^{40}$ according to the multivariate normal distribution $N(0, I)$. We also randomly picked a hyper plane $\boldsymbol{\lambda} \in I\!R^{40}$ using the same normal distribution, and assigned to the i'th point the label $y_i = sign(\boldsymbol{\lambda} \cdot \mathbf{x}_i)$. We divided the points into four groups according to their margin. The groups are numbered such that the first group contains the quarter of the points attaining the largest margin while the fourth group contains the quarter of the points attaining the smallest margin. We conducted five experiments. In each experiment a different subset of the groups was contaminated with label noise. In the first experiment no group was contaminated with label noise. In the second experiment only the first group was contaminated with a label noise distributed Bernoulli $B(p)$. In the third set of experiments we contaminated the first and the second group, and so on until all four groups were contaminated with label noise resulting in a uniform Bernoulli noise for all points. Since we contaminated only a subset of the points, the overall noise rate in experiment i was $p(i-1)/4$. In each experimental setup we compared the log-loss version of the parallel boosting algorithm from [2], to the LLM algorithm with ε set to be p, and to the LLD algorithm with μ set to $\ln((1-p)/p)$. To compare the performances of the algorithms we generated a test set containing 1000 points. The test set, in contrast to the train set, is noise-free. This is compatible with our assumption, in Sec. 2.1, that the data is actually linearly separable, but our train set was contaminated with label noise. We repeated this experimental setup with four different values of p, and each experiment was repeated ten times. Average results are presented in Fig. 4, in which each plot corresponds to a different value of p.

It is clear from the figure that boosting is sensitive to malicious noise which flips the labels of points that attain large margin values. This can be seen by the high increase of error caused by the contamination of the first quarter (high margin examples). As we contaminate the other quarters, the increase of error becomes much smaller. In contrast, LLM actually leverages from this type of noise. The error for LLM is very low when we contaminate only the first two quarters. Only when the last quarter is contaminated, there is a significant increase in the error. Even then, the overall error of LLM is much lower then that of boosting. The error of LLD increases linearly with the amount of contaminated quarters. Thus, LLD is less sensitive to the margin of the outliers than boosting, yet it is not as effective as LLM. The poor performance of boosting is not surprising – quite a few papers have formerly noted that boosting algorithms are highly sensitive to *label* noise (see for instance [5] and the references therein). This type of noise however is far less crucial for LLM and LLD as it conveys information on the examples whose labels are likely to be incorrect. The error rate of LLM and LLD increases as the noise becomes more uniform. Nonetheless, even with an i.i.d Bernoulli noise, LLM and LLD exhibit a far lower test error than boosting.

USPS: The USPS (US Postal Service) dataset is known to be a challenging classification task particularly since its training set and test set were collected in a different manner. The dataset contains $7,291$ training examples and $2,007$ test examples. Each example is represented as a 16×16 matrix where each entry in the matrix is a pixel that can take a value in $\{0, \ldots, 255\}$. Each example is associated with a label in $\{0, \ldots, 9\}$ which is the digit content of the image. We broke the 10-class problem into 10 binary problems. The i'th problem was to discriminate the digit i from the rest. For each binary problem we compared the test error of the following algorithms: log-loss boosting , LLM with fixed ε (at 0.08 and 0.16), LLM with variable ε (starting from 0.08 and modified every 100 approximate M-steps), and the LLD algorithm with $\mu = 4$. A comparative representation of these results can be seen in the left side of Fig. 5. In this figure we show the difference between the test error (percentage wise) of boosting and the test error of our algorithms. Each group of bars describes the results of one of our algorithms on each of the digits. We can see that after 100 boosting steps the LLD algorithm is the best performing algorithm. As we increase the number of iterations LLM, especially with a fixed value for ϵ, takes the charge. LLM clearly outperforms the boosting

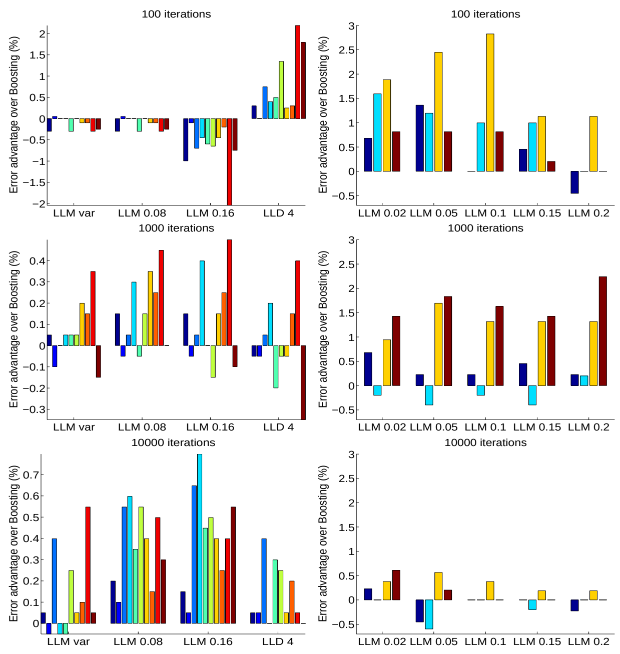

Figure 5. Comparison of the LLM and LLD algorithms with various parameters to the boosting algorithm of [2]. We use the parallel version on the USPS dataset (left) and sequential version on the UseNet data set (right). We show the difference between the boosting test error and the test error of our algorithms with various parameters.

algorithm. This may partially contributed to our margin loss function which is less prone to outliers, as discussed in Sec. 5. From Fig. 5 we see that LLM improves the boosting result at least by a small margin, and for some digits its error rate is lower than boosting's by as much as 0.8%.

UseNet: This dataset consists of Usenet articles collected by Lang [11] from 20 different newsgroups. One thousand articles were collected for each newsgroup so there are $20,000$ articles in the entire collection. In our experiments we randomly divided the articles from each newsgroup into a training set of 750 articles and a test set of 250 articles. The binary classification tasks we checked discriminate between articles from pairs of topics. Due to lack of space we show the results for only four pairs of newsgroups topics.

We compared log-loss boosting with the sequential version of LLM using single words as features. (The ith entry in a vector representing a document is 1 if the ith word appears in the document, and is 0 otherwise.) A comparison of the results obtained by LLM and boosting is given on the right side of Fig. 5. When the number of rounds is 100 or 1000 then LLM often outperforms boosting. The error rate of LLM in many cases is as much as 2% lower. As the number of rounds gets to $10,000$ the results of LLM and vanilla boosting become indistinguishable. One possible explanation to this type of behavior is that LLM finds better base-hypotheses (i.e. words) than boosting due to its improved criterion for choosing features. Alas, as the number of rounds grows, boosting has the opportunity to choose many features and thus close the gap. We plan to examine this angle more thoroughly in future research.

References

[1] P. L. Bartlett and S. Mendelson. Rademacher and Gaussian complexities: Risk bounds and structural results. COLT'01.

[2] M. Collins, R.E. Schapire, and Y. Singer. Logistic regression, AdaBoost and Bregman distances. *Machine Learning*, 47(2/3), 2002.

[3] O. Dekel, S. Shalev-Shwartz, and Y. Singer. Smooth epsilon-insensitive regression by loss symmetrization. COLT'03.

[4] A.P. Dempster, N.M. Laird, and D.B. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. of the Royal Statistical Society, Ser. B*, 39:1–38, 1977.

[5] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2), 1999.

[6] N. Duffy and D. Helmbold. Potential boosters ? NIPS'99.

[7] Y. Freund. Boosting a weak learning algorithm by majority. *Information and Computation*, 121(2), 1995.

[8] Y. Freund. An adaptive version of the boost by majority algorithm. COLT'99.

[9] Y. Freund and R. E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *J. of Computer and System Sciences*, 55(1), 1997.

[10] K. U. Höffgen, K. S. Van Horn, and H. U. Simon. Robust trainability of single neurons. *JCSS*, 50(1), 1995.

[11] K. Lang. Newsweeder: Learning to fi lter netnews. ICML'95.

[12] L. Mason, J. Baxter, P. Bartlett, and M. Frean. Functional gradient techniques for combining hypotheses. In *Advances in Large Margin Classifi ers*. 1999.

[13] K. Nigam, A. McCallum, S. Thrun, and T. Mitchell. Text classifi cation from labeled and unlabeled documents using em. *Machine Learning*, 39(2/3), 2000.

[14] G. Rätsch. *Robust Boosting and Convex Optimization*. Doctoral dissertation, University of Potsdam, 2001.

[15] R. E. Schapire and Y. Singer. Improved boosting algorithms using confi dence-rated predictions. *ML*, 37(3), 1999.

[16] R. A. Servedio. Smooth boosting and learning with malicious noise. COLT'01, EuroCOLT'01.

[17] H. Tuy. 1994. D.C. optimization: Theory, Methods & Algorithms. Horst & Pardalos (Ed) *Handbook of Global Opt.*

[18] S. Wang, R. Rosenfeld, Y. Zhao, and D. Schuurmans. The latent maximum entropy principle. In *IEEE International Symposium on Information Theory*, 2002.

[19] X. Zhang X.Shen, G. C. Tseng and W. H. Wong. On psi-learning.i. *J. of American Statistical Association*, 2003.